Posibilidad de varios periféricos operando mediante interrupciones.*

Juan Zamorano Flores

1 Introducción.

En clases previas se ha presentado el mecanismo básico de las interrupciones. Se ha definido cómo se solicitan, cuándo se atienden y qué es la rutina de servicio de la interrupción. Todo este mecanismo básico se ha ilustrado con un ejemplo completo de realización de una operación de entrada/salida mediante interrupciones para un periférico sencillo.

En esta clase se presentarán los problemas que se introducen en un sistema de entrada/salida con varios periféricos operando mediante interrupciones. Estos problemas son:

- Identificación y localización de las rutinas de servicio. Si existen varios periféricos que pueden solicitar interrupciones, es necesario identificar al peticionario para ejecutar su rutina de servicio. Como cada periférico tiene un funcionamiento distinto, debe existir una rutina de servicio para cada periférico y se trata de ejecutar la rutina de servicio correspondiente al que solicita el servicio de la CPU.

- Prioridades en caso de peticiones simultáneas. Otro problema que se plantea es qué hacer en caso de peticiones simultáneas. Es decir, puede darse el caso que la atención de la CPU sea solicitada a la vez por varios dispositivos. Este es un problema común en un computador: un recurso único que se solicita simultáneamente por varios peticionarios. La solución consiste en asignar prioridades a los peticionarios y, en caso de peticiones simultáneas, el recurso se concede al peticionario de mayor prioridad. En este caso, el recurso es la atención de la CPU y los peticionarios los dispositivos periféricos. Los criterios para asignar prioridades a los dispositivos periféricos se explicarán en la siguiente clase.

- Anidamiento de rutinas de servicio. Las prioridades indican una urgencia de ejecución. De esta manera, existen dispositivos con una urgencia mayor que otros, lo que se persigue es que los dispositivos con mayor urgencia puedan interrumpir el servicio de los de menor urgencia.

Es decir, los periféricos de mayor prioridad deben poder interrumpir la rutina de servicio de los periféricos de menor prioridad, pero no al revés.

Naturalmente, existe un problema previo de conexionado: ¿Cómo se conectan varios periféricos a la única línea que posee la CPU de petición de interrupciones ($\overline{INT}$)? Los periféricos se conectan todos a esa línea con una salida en colector abierto. Al ser la línea activa a nivel bajo, la línea se activa en el momento que sólo uno la active y permanecerá activa hasta que todos los periféricos hayan desactivado su petición.

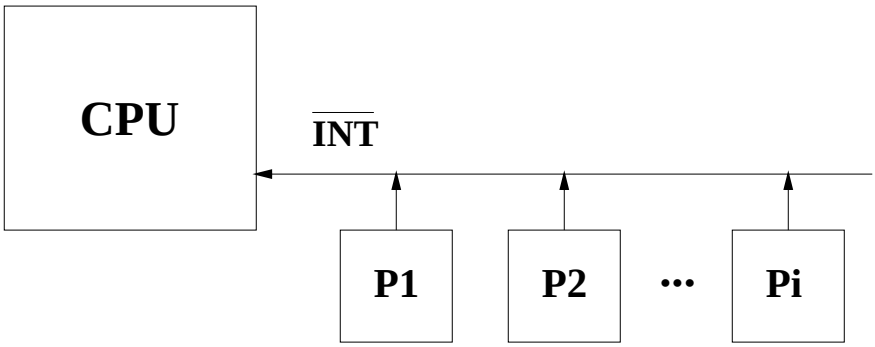

Figura 1: Conexión de varios periféricos a la línea de petición de interrupciones.

2 Solución mediante identificación por muestreo (*polling*).

Con el mecanismo básico que hemos definido, la atención a una interrupción supone la ejecución de una rutina de servicio de interrupciones que está situada en una posición predefinida de memoria.

Según se explicó en clases anteriores, las interrupciones se atienden al final de cada instrucción y se había redefinido la secuencia de *fetch* para tener en cuenta la solicitud de interrupciones:

Recordemos también la secuencia de operaciones de la instrucción de retorno de interrupción:

Es decir, con este mecanismo básico sea cual sea el peticionario de la interrupción se ejecuta una única rutina de tratamiento de interrupciones.

2.1 Identificación por muestreo.

Con el esquema de interrupciones descrito se puede identificar al peticionario de la interrupción por software. Es decir, en la rutina única se procede a muestrear uno a uno todos los posibles peticionarios hasta encontrar al solicitante de la interrupción.

De esta forma debemos leer los registros de control/estado de los módulos de entrada/salida de cada periférico para comprobar si tienen una petición pendiente. Naturalmente, es requisito indispensable que los módulos reflejen esta contingencia. A continuación, mediante una instrucción de bifurcación se ejecuta la rutina de tratamiento específica de cada periférico.

```
FETCH: Si INT ∧ SR.BMI entonces:
          Salvar PC
          Salvar SR
          SR.BMI ← 1 (Inhibir interrupciones)
          SR.S ← 1 (Modo privilegiado)
          PC ← DRTI (Dir.  de la rutina de tratamiento de int.)
          ir a fetch
       Si no
          ir a fetch
```

Figura 2: Secuencia básica de reconocimiento de interrupciones.

```
RETI: Restaurar SR
      Restaurar PC
      ir a FETCH
```

Figura 3: Instrucción de retorno de interrupción.

Una posible codificación de la secuencia de muestreo software se presenta en la figura 4.

```
DRTI:  PUSH   .R1
       IN     .R1, /Dir_C/E_P1
       CMP    .R1, #INT_Pendiente_P1
       BZ     $Rut_Espec_P1
       IN     .R1, /Dir_C/E_P2
       CMP    .R1, #INT_Pendiente_P2
       BZ     $Rut_Espec_P2
       ...

Rut_Espec_P1:Salvar estado
             Código
             específico
             de P1
             Restaurar estado
             RETI

Rut_Espec_P2:Salvar estado
             Código
             específico
             de P2
             Restaurar estado
             RETI

             ...
```

Figura 4: Codificación de la secuencia de muestreo.

2.2 Esquema de prioridades software.

Este mecanismo de identificación por muestreo soluciona el problema de peticiones simultáneas. En efecto, en caso de petición de interrupción se ejecuta la rutina única de reconocimiento y la

secuencia de muestreo supone una asignación de prioridades.

En el caso de la figura 4, el periférico P1 tiene asignada una mayor prioridad, y en caso de petición de P1 y Pi se ejecuta primero la rutina específica de P1. En esa rutina se desactivará la petición de P1 pero no la de Pi, de este modo cuando se ejecuta la instrucción `RETI` las interrupciones vuelven a estar permitidas y se ejecuta de nuevo la secuencia de reconocimiento de interrupciones. Así se ejecuta de nuevo la secuencia de muestreo y se detecta que Pi tiene una interrupción pendiente con lo que se ejecutará la rutina específica de este periférico.

De este modo, el orden de muestreo supone una asignación de prioridades, dándose mayor prioridad a los periféricos a los que se le consulta en primer lugar. Por lo tanto, en caso de peticiones simultáneas, el periférico que esté antes en la secuencia será atendido primero.

2.3 Anidamiento de rutinas de servicio.

Sin embargo, con este mecanismo básico no se puede realizar un anidamiento correcto de rutinas de servicio. La premisa establece que periféricos más prioritarios puedan interrumpir el servicio de los periféricos menos prioritarios pero nunca al revés.

En primer lugar, durante la secuencia de reconocimiento de interrupciones se han enmascarado las interrupciones, con lo cual para permitir que las rutinas de servicio sean interrumpidas se ha de colocar una instrucción `EI`. Se puede caer en la tentación de colocar estas instrucciones en todas las rutinas de servicio de todos periféricos, con la excepción de la correspondiente al más prioritario. Naturalmente deben colocarse tras la desactivación de la solicitud, de otro modo se reconocería la misma interrupción indefinidamente.

De este modo, P1 (el más prioritario en el ejemplo) podría interrumpir el servicio de cualquier otro periférico pero no al revés. Es decir, se cumpliría la premisa para P1, pero ¿qué ocurre con los demás?. Simplemente que cualquiera podría interrumpir a cualquiera, ya que con este esquema la identificación del peticionario es posterior al reconocimiento de la interrupción. Es decir, la interrupción debe ser reconocida para poder averiguar mediante software la prioridad del peticionario. Por lo tanto, no se cumple la premisa, ya que periféricos menos prioritarios interrumpirían el servicio de periféricos más prioritarios y el anidamiento de rutinas no sería correcto.

3 Identificación mediante vectorización.

Una mejora sobre el mecanismo de identificación descrito en la sección anterior consiste en relegar a los periféricos la tarea de la identificación. De este modo, el tiempo empleado en la secuencia de muestreo se puede dedicar a avanzar en la ejecución de otros programas. A esta técnica se le denomina vectorización y son los propios peticionarios de la interrupción los que se identifican cuando se lo solicita la CPU.

Para ello se necesita una nueva señal de control, la señal de reconocimiento de interrupciones ($\overline{INTA}$, *INTerrupt Acknowledge*). El cometido de esta señal de control es marcar la temporización de un ciclo especial de bus cuyo objetivo es leer un identificador del periférico que solicita la interrupción. Se trata pues de un ciclo especial de lectura.

En la figura 5 se observa un cronograma simplificado de este ciclo de bus:

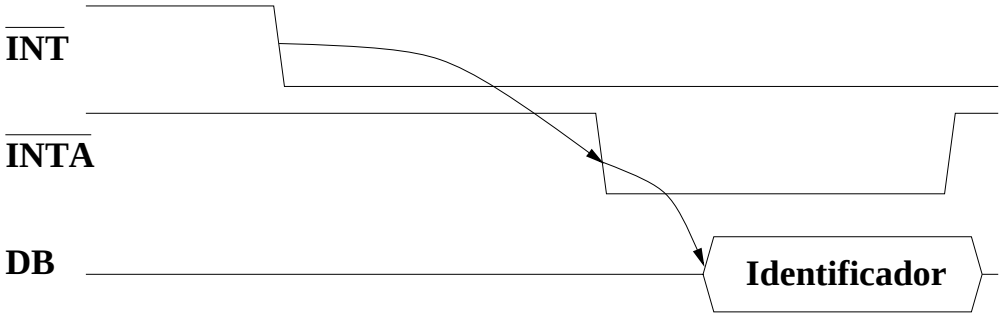

Figura 5: Ciclo de bus de reconocimiento de interrupciones.

Como consecuencia de la activación de la señal $\overline{INT}$, la CPU lanza un ciclo de lectura especial caracterizado por la activación de la señal $\overline{INTA}$ y cuyo objetivo es leer un identificador del periférico. Este identificador recibe el nombre de vector de interrupción.

Cada periférico tiene asignado un vector distinto, que se almacena en un nuevo registro del módulo de entrada/salida: el registro del vector de interrupción. Por lo tanto, la rutina de inicialización del computador se debe encargar de inicializar estos registros de modo que cada periférico proporcione el vector adecuado.

3.1 Secuencia de reconocimiento con vectorización.

La secuencia de reconocimiento de interrupciones es diferente en el caso de que la CPU disponga del mecanismo de identificación mediante vectorización:

```
FETCH: Si INT ∧ SR.BMI entonces:
          Salvar PC
          Salvar SR
          SR.BMI ← 1 (Inhibir interrupciones)
          SR.S ← 1 (Modo privilegiado)
          Ciclo de bus de reconocimiento de interrupciones
          (activación INTA) DR ← Vector
          PC ← f(Vector)
          ir a fetch
       Si no
          ir a fetch
```

Figura 6: Secuencia de reconocimiento de interrupciones con vectorización.

En este caso, se lanza un ciclo de bus de reconocimiento de interrupciones con el objetivo de leer el vector de interrupción. Este vector es el identificador proporcionado por el periférico y a partir de él se obtiene la dirección de inicio de la rutina de tratamiento de la interrupción específica

del periférico. Nótese, que la secuencia de operaciones correspondientes a la instrucción de retorno de interrupción, sigue siendo la de la figura 3.

La dirección de comienzo de la rutina de tratamiento de la interrupción se obtiene aplicando una función al vector proporcionado por el periférico. Esta función puede ser la identidad, es decir, el periférico proporciona la dirección de la rutina de tratamiento. Pero, lo más habitual es que esta función sea un direccionamiento relativo a registro base siendo el vector el desplazamiento. El registro base es un registro especial que llamaremos Registro Base de la Tabla de Vectores (RBTV) y que apunta a una estructura que contiene la dirección de comienzo de todas las rutinas de tratamiento de interrupción. Esta estructura recibe el nombre de Tabla de Vectores de Interrupción.

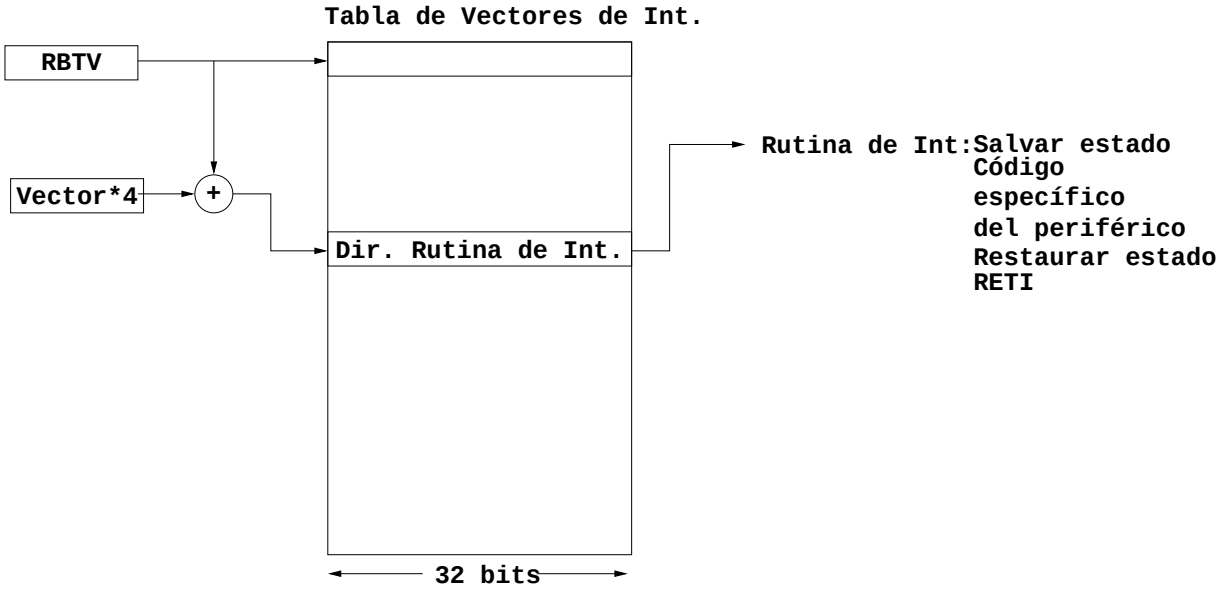

Figura 7: Obtención de la dirección de inicio de la rutina a través de la tabla de vectores.

La figura 7 muestra gráficamente la forma de obtener la dirección de inicio de la rutina de tratamiento a partir del vector de interrupción para un computador con direcciones de 32 bits y direccionamiento a nivel de byte.

3.2 Esquema de prioridades hardware.

Mediante el mecanismo de vectorización se identifica el periférico solicitante de la interrupción al recibir la señal $\overline{INTA}$. Sin embargo, en caso de peticiones simultáneas sólo se debe identificar el periférico más prioritario de los que han solicitado interrupción.

Por lo tanto, se trata de conseguir que la señal $\overline{INTA}$ llegue a uno y sólo a uno de los periféricos que han solicitado interrupción. Además, ese periférico deberá ser el más prioritario de todos los que han solicitado la interrupción.

Existen varios esquemas hardware para gestionar la señal $\overline{INTA}$ de forma que sólo llegue al periférico más prioritario.

3.2.1 Gestión centralizada.

Este esquema se basa en un gestor centralizado de prioridades. Este gestor tiene un interfaz hacia a la CPU y otro hacia los periféricos.

La conexión a la CPU se realiza a través de las dos líneas de petición y reconocimiento de interrupciones ($\overline{INT}, \overline{INTA}$), tal como se aprecia en la figura 8. El interfaz hacia los periféricos se compone de n parejas de líneas de petición y reconocimiento de interrupciones ($\overline{INT_i}, \overline{INTA_i}$), a cada par de líneas se conecta un único periférico. Además, cada par de líneas tiene asignada una prioridad distinta.

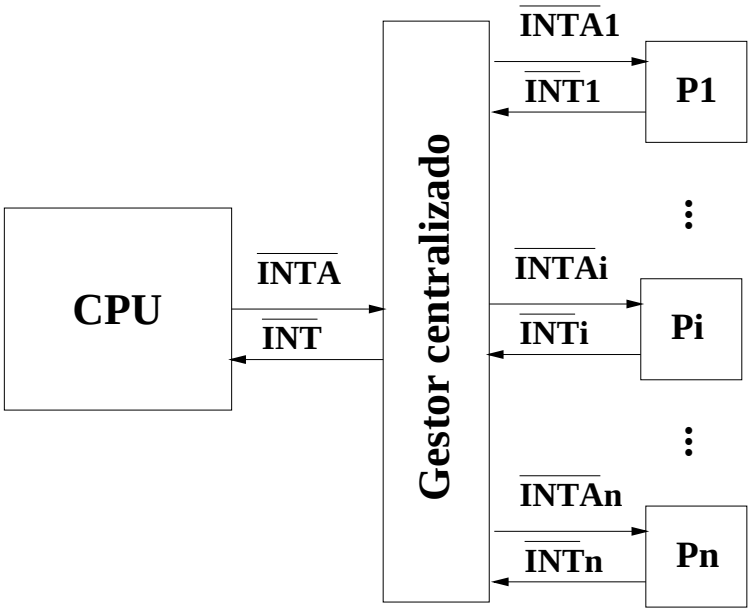

Figura 8: Gestión centralizada de prioridades.

El comportamiento del gestor centralizado es el siguiente:

- Si se activa una o más líneas de petición ($\overline{INT_i}$), el gestor activa la línea de petición de interrupciones hacia la CPU ($\overline{INT}$).
- Cuando la CPU reconoce la interrupción y lanza el ciclo de bus de reconocimiento de interrupciones activando $\overline{INTA}$, el gestor activa la línea $\overline{INTA_i}$ correspondiente a la petición de mayor prioridad ($\overline{INT_i}$).

De esta forma, el periférico P_i se identifica colocando su vector de interrupción cuando recibe la señal $\overline{INTA_i}$.

Este esquema tiene el inconveniente de no ser ampliable, ya que si se dispone de n parejas de líneas sólo se pueden conectar n periféricos.

3.2.2 Gestión encadenada o *daisy chain*.

Con este esquema los periféricos se conectan todos a la línea de petición de interrupciones ($\overline{INT}$), y la línea de reconocimiento de interrupciones ($\overline{INTA}$) se conecta de forma encadenada a través de todos (figura 9).

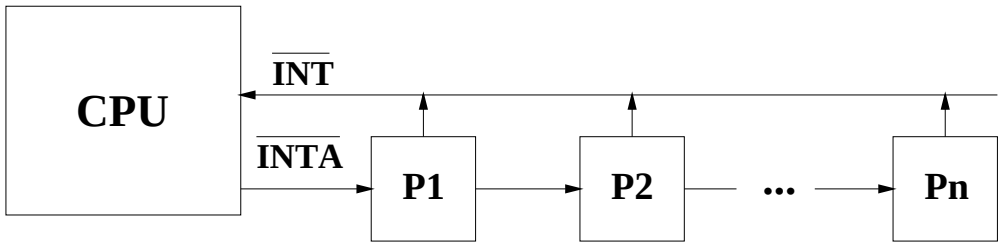

Figura 9: Gestión encadenada de prioridades.

El funcionamiento de este esquema es el siguiente:

- La línea de petición de interrupciones ($\overline{INT}$) se activa siempre que uno o más periféricos coloquen a nivel bajo su salida en colector abierto.
- Cuando la CPU reconoce la interrupción y lanza el ciclo de bus de reconocimiento de interrupciones activando $\overline{INTA}$, esta señal se propaga a través de la cadena como si de un testigo se tratase. De modo que, si el testigo llega a un periférico que no ha solicitado interrupción, éste lo propaga al siguiente de la cadena. Por contra, si llega a un periférico que sí ha solicitado interrupción, el periférico en cuestión se queda el testigo y coloca en el bus de datos su vector de interrupción.

De esta forma, únicamente el periférico más prioritario es el que se identifica. La prioridad es función de la posición que ocupan los periféricos en la cadena, siendo, en la figura 9, P_1 el de mayor prioridad y P_n el menos prioritario.

A diferencia del esquema centralizado, este esquema es fácilmente ampliable, se pueden conectar tantos periféricos como se necesiten. Sin embargo, el ciclo de bus de reconocimiento de interrupciones debe tener la duración suficiente (incluyendo ciclos de espera si es necesario) para que el testigo ($\overline{INTA}$) pueda propagarse hasta el final de la cadena. Por lo tanto, este esquema es lento si existen muchos periféricos en la cadena.

3.2.3 Gestión híbrida.

Como se ha visto, las ventajas e inconvenientes de los esquemas anteriores son contrapuestos. Por lo tanto, una forma de obtener las ventajas de ambos esquemas sin padecer sus inconvenientes es conjugar ambos en un esquema híbrido.

Ya que el inconveniente de la gestión centralizada es que no es ampliable, colocamos de forma encadenada varios periféricos a cada par de líneas ($\overline{INT_i}, \overline{INTA_i}$) de un gestor centralizado tal como se aprecia en la figura 10.

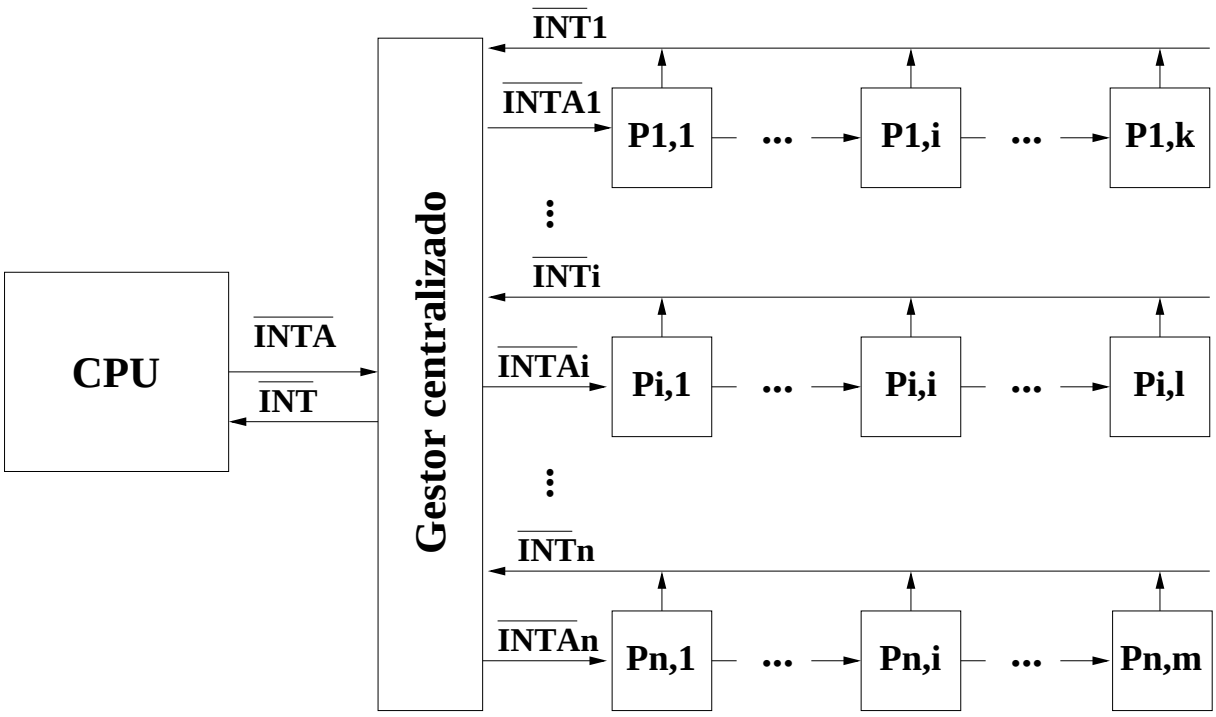

Figura 10: Gestión híbrida de prioridades.

Así, tenemos un esquema ampliable ya que disponemos de n esquemas encadenados. Además, esta gestión híbrida no padece de la lentitud de la encadenada puesto que al tener n cadenas, la longitud de cada una será considerablemente menor.

3.3 Anidamiento de rutinas de servicio.

Mediante la identificación por vectorización y un esquema de prioridades hardware como los descritos, se resuelven dos de los tres problemas planteados cuando existen varios periféricos operando mediante interrupciones. Por lo tanto, queda por resolver el del anidamiento de rutinas de servicio.

El problema reside en que sólo se dispone de un biestable para enmascarar las interrupciones. De este modo, o bien tenemos todas las interrupciones permitidas o bien todas prohibidas. Como lo que se pretende es discernir entre varios niveles de prioridad y, de este modo, atender las solicitudes de niveles más prioritarios interrumpiendo el servicio de peticiones de menor prioridad y no al revés, la solución consiste en poseer varios niveles de prioridad y prohibir la atención a peticiones de menor prioridad mientras se ejecuta la rutina de servicio de una petición más prioritaria.

Es decir, en vez de un solo biestable de máscara de interrupción (BMI) se dota al registro de estado (SR) de varios, de forma que se codifica en ellos el nivel de la interrupción que se está atendiendo. De forma que no se atienden interrupciones de nivel menor o igual al codificado en la máscara de interrupción (MI) y así se dota a la CPU de un mecanismo de inhibición selectiva por niveles de prioridades de interrupción.

Naturalmente, para conocer el nivel de la interrupción que se solicita son necesarias a su vez varias líneas de petición de interrupción ($\overline{INT_i}$). Además, también son necesarias varias líneas de reconocimiento ($\overline{INTA_i}$), puesto que podrá haber peticiones simultáneas de varios niveles y sólo se atenderá la de mayor de prioridad de todas ellas (siempre y cuando el nivel de la petición sea mayor que el almacenado en la máscara de interrupción). Por ejemplo, con 3 biestables de máscara de interrupción se codifican 8 niveles de prioridad y se requieren 7 parejas de líneas de petición y reconocimiento de interrupciones.

La secuencia de reconocimiento de interrupciones para un computador con inhibición por niveles de interrupción e identificación por vectorización sería la expuesta en la figura 11.

```
FETCH: Si (i = max(INT_j), ∀j) ∧ (i > SR.MI) entonces:
          Salvar PC
          Salvar SR
          SR.MI ← i (Establecer la nueva máscara de int.)
          SR.S ← 1 (Modo privilegiado)
          Ciclo de bus de reconocimiento de interrupciones
          (activación INTA_i) DR ← Vector
          PC ← f(Vector)
          ir a fetch
       Si no
          ir a fetch
```

Figura 11: Secuencia de reconocimiento de interrupciones con vectorización e inhibición por niveles.

Según establece la condición, sólo se ejecuta la secuencia de reconocimiento de interrupciones si la petición de interrupción de mayor prioridad es más prioritaria que la que se está atendiendo, cuyo nivel se codifica en la máscara de interrupción del registro de estado (SR.MI). Nótese, que la secuencia de operaciones correspondientes a la instrucción de retorno de interrupción (`RETI`), sigue siendo la de la figura 3.

3.3.1 Ejemplo.

Para ilustrar el funcionamiento de este mecanismo su utiliza un ejemplo de anidamiento de rutinas de servicio. En la figura 12 se muestra la secuencia de ejecución de estas rutinas, así como la máscara de interrupción del registro de estado (`SR.MI`) durante la ejecución de cada rutina y las máscaras almacenadas en la pila.

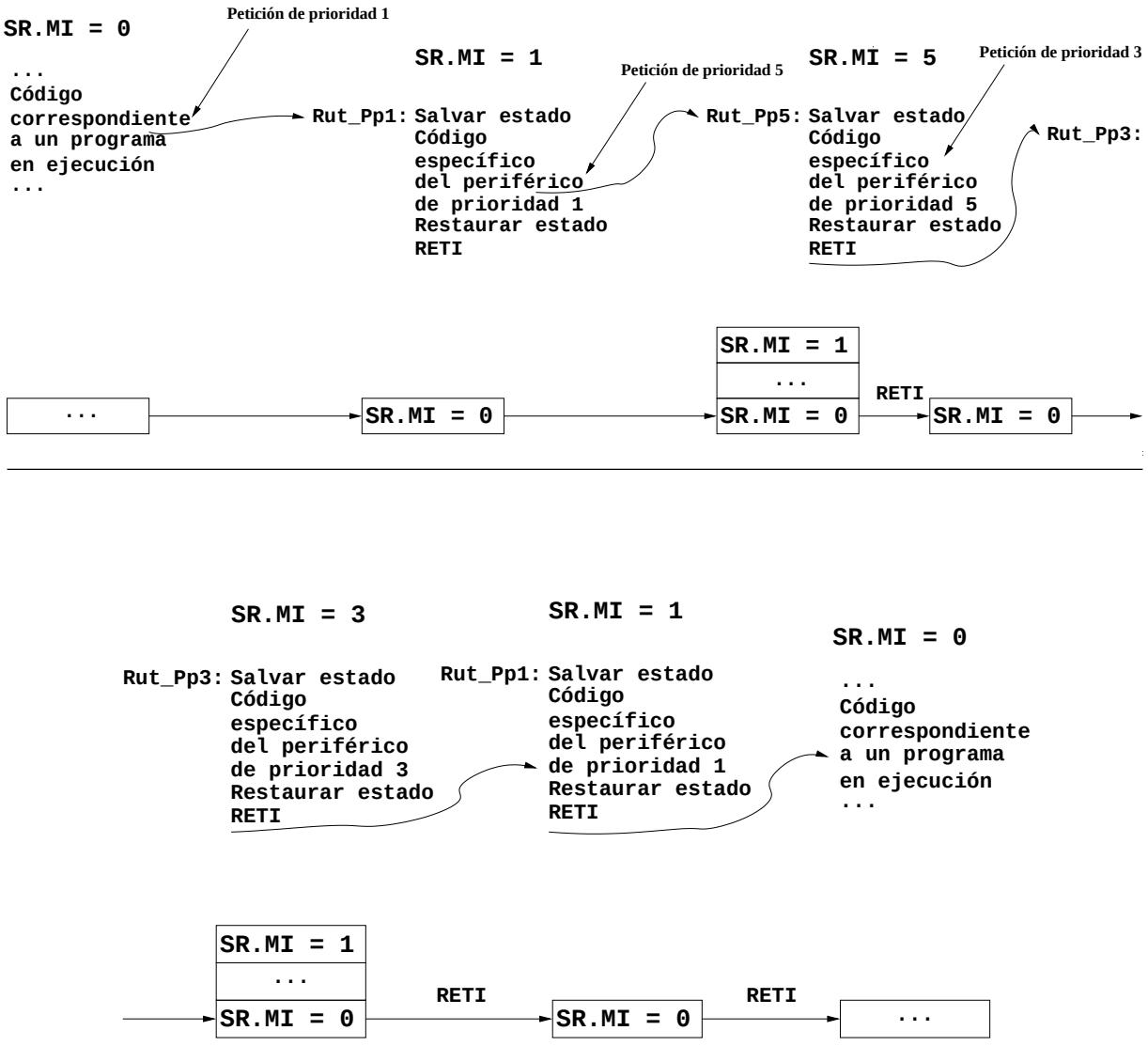

Figura 12: Anidamiento de rutinas de servicio de interrupción.

En la situación de partida existe un programa en ejecución y la máscara de interrupciones del registro de estado (`SR.MI`) contiene un 0, es decir están permitidas todas las interrupciones. A continuación un periférico de prioridad 1 solicita una interrupción, cuando acabe la ejecución de esa instrucción en curso, se ejecuta la secuencia de `FETCH` y como el nivel de la interrupción es superior a la máscara se reconoce la interrupción. Como consecuencia, la máscara anterior correspondiente al programa (`SR.MI=0`) se almacena en la pila, se actualiza la máscara del registro de estado (`SR.MI=1`) y se bifurca a la rutina correspondiente (`Rut_Pp1`).

Durante la ejecución de esta rutina, un periférico de prioridad 5 solicita una interrupción. Como nivel de la interrupción es superior a la máscara se reconoce esta interrupción. La máscara

correspondiente a la rutina (SR.MI=1) se almacena en la pila, se actualiza la máscara del registro de estado (SR.MI=5) y se bifurca a la rutina correspondiente (Rut_Pp5).

Posteriormente, durante la ejecución de la rutina de servicio del periférico de prioridad 5 se solicita una interrupción de prioridad 3. En este caso no se reconoce la interrupción, ya que la prioridad de la interrupción es inferior al contenido de la máscara del registro de estado (SR.MI=5). De este modo, se impide que un periférico de menor prioridad (3) interrumpa el servicio de otro de mayor prioridad (5).

Esta solicitud de interrupción de nivel 3 será atendida cuando finalice la rutina de servicio y se ejecute la instrucción RETI. Esta instrucción restaura el registro de estado almacenado en la pila (SR.MI=1) y bifurca a la secuencia de FETCH para ejecutar la siguiente instrucción. Por lo tanto, se ejecuta la secuencia de reconocimiento de interrupciones ya que la prioridad de la petición es 3 y la máscara contiene un 1.

Una vez se ejecuta la rutina de tratamiento del periférico de prioridad 3 (Rut_Pp3), la instrucción RETI restaura el registro de estado y el contador de programa con lo que se reanuda la ejecución de la rutina Rut_Pp1. De la misma forma, al acabar la ejecución de Rut_Pp1 se reanuda la ejecución del programa interrumpido.

4 Conclusiones.

En esta clase se han expuesto los problemas que surgen cuando existen varios periféricos operando mediante interrupciones. Mediante el mecanismo básico de interrupciones se pueden abordar estos problemas, con la excepción del problema de anidamiento de rutinas de servicio.

Este mecanismo básico, tiene el inconveniente de que el proceso de identificación del peticionario de la interrupción recae en la CPU, que debe ejecutar una secuencia de muestreo para identificar al periférico y ejecutar la rutina correspondiente. Mediante la vectorización, es el periférico peticionario el que se identifica cuando se lo requiere la CPU. Para ello se necesita una señal de control específica que caracteriza el ciclo de bus de reconocimiento de interrupciones.

En este caso, se requiere un esquema de prioridades hardware para conducir esta señal al periférico más prioritario que solicitó una interrupción. Existen varios esquemas posibles y se han estudiado las ventajas e inconvenientes de cada uno de ellos.

Por último, es necesario un mecanismo de inhibición selectiva por niveles de prioridad para realizar un anidamiento de rutinas de servicio. Este mecanismo se basa en tener varios biestables de máscara de interrupción para almacenar el nivel de la interrupción que se está atendiendo.